J 6

```
MMM         MMM  TTTTTTTTTTTTTTT     AAAAAAAAA        AAAAAAAAA        CCCCCCCCCCCC  PPPPPPPPPPPP
MMM         MMM  TTTTTTTTTTTTTTT     AAAAAAAAA        AAAAAAAAA        CCCCCCCCCCCC  PPPPPPPPPPPP
MMM         MMM  TTTTTTTTTTTTTTT     AAAAAAAAA        AAAAAAAAA        CCCCCCCCCCCC  PPPPPPPPPPPP
MMMMMM   MMMMMM        TTT        AAA         AAA  AAA         AAA  CCC              PPP         PPP
MMMMMM   MMMMMM        TTT        AAA         AAA  AAA         AAA  CCC              PPP         PPP
MMMMMM   MMMMMM        TTT        AAA         AAA  AAA         AAA  CCC              PPP         PPP
MMM   MMM   MMM        TTT        AAA         AAA  AAA         AAA  CCC              PPP         PPP
MMM   MMM   MMM        TTT        AAA         AAA  AAA         AAA  CCC              PPP         PPP
MMM   MMM   MMM        TTT        AAA         AAA  AAA         AAA  CCC              PPP         PPP
MMM         MMM        TTT        AAA         AAA  AAA         AAA  CCC              PPPPPPPPPPPP
MMM         MMM        TTT        AAA         AAA  AAA         AAA  CCC              PPPPPPPPPPPP
MMM         MMM        TTT        AAA         AAA  AAA         AAA  CCC              PPPPPPPPPPPP
MMM         MMM        TTT        AAAAAAAAAAAAAAA  AAAAAAAAAAAAAAA  CCC              PPP
MMM         MMM        TTT        AAAAAAAAAAAAAAA  AAAAAAAAAAAAAAA  CCC              PPP
MMM         MMM        TTT        AAAAAAAAAAAAAAA  AAAAAAAAAAAAAAA  CCC              PPP
MMM         MMM        TTT        AAA         AAA  AAA         AAA  CCC              PPP
MMM         MMM        TTT        AAA         AAA  AAA         AAA  CCC              PPP
MMM         MMM        TTT        AAA         AAA  AAA         AAA  CCC              PPP
MMM         MMM        TTT        AAA         AAA  AAA         AAA     CCCCCCCCCCCC  PPP
MMM         MMM        TTT        AAA         AAA  AAA         AAA     CCCCCCCCCCCC  PPP
MMM         MMM        TTT        AAA         AAA  AAA         AAA     CCCCCCCCCCCC  PPP
```

```
**FILE**ID**COMLABPRC                              N 2



  CCCCCCCC    OOOOOO    MM      MM  LL            AAAAAA    BBBBBBBB    PPPPPPPP    RRRRRRRR      CCCCCCCC
  CCCCCCCC    OOOOOO    MM      MM  LL            AAAAAA    BBBBBBBB    PPPPPPPP    RRRRRRRR      CCCCCCCC
CC          OO      OO  MMMM  MMMM  LL          AA      AA  BB      BB  PP      PP  RR      RR  CC
CC          OO      OO  MMMM  MMMM  LL          AA      AA  BB      BB  PP      PP  RR      RR  CC
CC          OO      OO  MM  MM  MM  LL          AA      AA  BB      BB  PP      PP  RR      RR  CC
CC          OO      OO  MM  MM  MM  LL          AA      AA  BB      BB  PP      PP  RR      RR  CC
CC          OO      OO  MM      MM  LL          AA      AA  BBBBBBBB    PPPPPPPP    RRRRRRRR    CC
CC          OO      OO  MM      MM  LL          AA      AA  BBBBBBBB    PPPPPPPP    RRRRRRRR    CC
CC          OO      OO  MM      MM  LL          AAAAAAAAAA  BB      BB  PP          RR  RR      CC
CC          OO      OO  MM      MM  LL          AAAAAAAAAA  BB      BB  PP          RR  RR      CC
CC          OO      OO  MM      MM  LL          AA      AA  BB      BB  PP          RR    RR    CC              ....
CC          OO      OO  MM      MM  LL          AA      AA  BB      BB  PP          RR    RR    CC              ....
  CCCCCCCC    OOOOOO    MM      MM  LLLLLLLLLL  AA      AA  BBBBBBBB    PP          RR      RR    CCCCCCCC    ....
  CCCCCCCC    OOOOOO    MM      MM  LLLLLLLLLL  AA      AA  BBBBBBBB    PP          RR      RR    CCCCCCCC    ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

```
0 MODULE COMLABPROC (LANGUAGE (BLISS32) ,
0                  IDENT = 'V04-000'
0                  ) =
1 BEGIN
1 !
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1
1 !++
1 !
1 ! FACILITY: INITIALIZE, MOUNT, MTAACP
1 !
1 ! ABSTRACT:
1 !       This module contains routines that are shared amoung the
1 !       MOUNT, INIT, and MTAACP. These routines deal with the
1 !       processing of the various labels that the MTAACP supports.
1 !
1 !
1 ! ENVIRONMENT:
1 !
1 !
1 !       VMS operating system, including privileged system services
1 !       and internal exec routines.
1 !
1 !--
1 !
1 !
1 !
1 ! AUTHOR:  Meg Dumont,                    CREATION DATE:  21-Feb-1983
1 !
1 ! MODIFIED BY:
1 !
1 !       V03-005 HH0041          Hai Huang       24-Jul-1984
1 !               Remove REQUIRE 'LIBD$:[VMSLIB.OBJ]MOUNTMSG.B32'.
1 !
1 !       V03-004 MMD0272         Meg Dumont,     23-Mar-1984  9:41
1 !               Add the common routine GET_RECORD part of support for $MTACCESS
1 !
```

```
1 !       V03-003 MMD0175          Meg Dumont,     26-May-1983  15:10
1 !               Change VOL1 to indicate ANSI level 4 when writing system
1 !               code in VOL1
1 !
1 !       V03-002 MMD0137          Meg Dumont,     12-Apr-1983  17:30
1 !               In TAPE_OWNER_PROT, added a check for a nonVMS nonblank
1 !               VOL1 OWNER IDENTIFIER field.
1 !
1 !       V03-001 MMD0122          Meg Dumont,     29-Mar-1983  0:46
1 !               This module is does the common ANSI label processing for
1 !               thE MTAACP, MOUNT and INIT.
1 !
1 !
1 !**
1
1 LIBRARY 'SYS$LIBRARY:LIB.L32';
1
1 REQUIRE 'SRC$:MTADEF.B32';
1
1 REQUIRE 'LIBD$:[VMSLIB.OBJ]INITMSG.B32';
1
1 FORWARD ROUTINE
1     GET_RECORD,                            ! routine to get record tape is reading
1     CHECK_PROT,                            ! check VMS protection on tape
1     FORMAT_VOLOWNER     : NOVALUE,         ! format the volume onwer field
1     PROCESS_VOL2_LABEL,                    ! intepret the VOL2 label
1     TAPE_OWN_PROT;                         ! determine the VMS owner and
1                                            ! protection of a tape
1     EXTERNAL ROUTINE
1         LIB$CVT_OTB     : ADDRESSING_MODE (GENERAL);
1
1
```

```
;   91          0604  1 GLOBAL ROUTINE GET_RECORD(UCB) =
;   92          0605  1
;   93          0606  1 !++
;   94          0607  1 !
;   95          0608  1 ! FUNCTIONAL DESCRIPTION:
;   96          0609  1 !       This routine is called before and after the call to $MTACCESS to return
;   97          0610  1 !       the record that the tape drive is currently processing
;   98          0611  1 !
;   99          0612  1 ! CALLING SEQUENCE:
;  100          0613  1 !       KERNEL_CALL (GET_RECORD, ARG1)
;  101          0614  1 !
;  102          0615  1 ! INPUT PARAMETERS:
;  103          0616  1 !       ARG1 -  Address of tapes UCB
;  104          0617  1 !
;  105          0618  1 ! IMPLICIT INPUTS:
;  106          0619  1 !       NONE
;  107          0620  1 !
;  108          0621  1 ! OUTPUT PARAMETERS:
;  109          0622  1 !       NONE
;  110          0623  1 !
;  111          0624  1 ! IMPLICIT OUTPUTS:
;  112          0625  1 !
;  113          0626  1 ! ROUTINE VALUE:
;  114          0627  1 !       Current record the tape drive is processing.
;  115          0628  1 !
;  116          0629  1 ! SIDE EFFECTS:
;  117          0630  1 !       NONE
;  118          0631  1 !
;  119          0632  1 ! USER ERRORS:
;  120          0633  1 !       NONE
;  121          0634  1 !
;  122          0635  1 !--
;  123          0636  1
;  124          0637  2     BEGIN
;  125          0638  2         MAP UCB : REF BBLOCK;
;  126          0639  2         RETURN .UCB[UCB$L_RECORD];
;  127          0640  1     END;


                                                                .TITLE  COMLABPROC
                                                                .IDENT  \V04-000\

                                                                .EXTRN  LIB$CVT_OTB

                                                                .PSECT  $CODE$,NOWRT,2

                                              0000 00000        .ENTRY  GET_RECORD, Save nothing                   ; 0604
                                 50      04  AC  D0 00002        MOVL    UCB, R0                                    ; 0639
                                 50    00B0  C0  D0 00006        MOVL    176(R0), R0                                ;
                                                04 0000B        RET                                                ; 0640

; Routine Size:  12 bytes,    Routine Base:  $CODE$ + 0000


;  128          0641  1
;  129          0642  1 GLOBAL ROUTINE TAPE_OWN_PROT ( VOLUIC, VOLUME_PROT : REF BITVECTOR[%BPVAL],
;  130          0643  1                                  PROCESS_UIC, VOL1 ) =
!++
!
! FUNCTIONAL DESCRIPTION:
!       This routine determines the tape owner and protection of the volume.
!       It uses the OWNER IDENTIFIER field of the VOL1 label. If this
!       field contians a value that VMS does not interpret then, the
!       user is required to have privledges to mount the tape. Unless
!       that user is the VMS owner of the tape determined from the VOL2
!       label.
!
! CALLING SEQUENCE:
!       TAPE_OWN_PROT ( ARG1, ARG2, ARG3, ARG4 )
!
! INPUT PARAMETERS:
!       ARG1 -  Address of area to store the volume uic
!       ARG2 -  Address of area to store the volume protection
!       ARG3 -  Process UIC
!       ARG4 -  Address of ANSI VOL1 label
!
! IMPLICIT INPUTS:
!       NONE
!
! OUTPUT PARAMETERS:
!       NONE
!
! IMPLICIT OUTPUTS:
!       VOLUME_UIC - owning uic of tape
!       VOLUME_PROT -  tape protection
!
! ROUTINE VALUE:
!       TRUE - Field was blank or was specied in VMS format
!       FALSE - Field was not VMS format, but was pre ANSI Label Standard
!               version 4 and the tape was created on another DEC operating
!               system that is it has D% information.
!
! SIDE EFFECTS:
!       NONE
!
! USER ERRORS:
!       NONE
!
!--

    BEGIN

    BIND
        VOLUME_UIC = .VOLUIC;                   ! Address of volume uic

    MAP
        VOL1    : REF BBLOCK;                   ! Address of VOL1 label

    LOCAL
        CONV_BUF        : VECTOR [6, BYTE],     ! buffer used for converting UIC
        VALUE,                                  ! used to hold parital UIC's
        P;                                      ! ptr into VOL1 tape owner field
```

```
2    ! bit numbers for different protections
2    !
2    LITERAL
2        WORLD_WRITE = 13,
2        WORLD_READ  = 12,
2        GROUP_WRITE = 9,
2        GROUP_READ  = 8;
2
2    ! If the LABEL STANDARD VERSION of the VOL1 label (CP 80) is a 4 then
2    ! do not process the VOL1 OWNER IDENTIFIER field.
2
2    IF .VOL1[VL1$B_LBLSTDVER] EQL '4'
2        THEN RETURN TRUE;
2
2    ! if ANSI tape produced by VAX system, decode tape owner field
2
2    IF .(VOL1[VL1$T_VOLOWNER])<0, 24> EQL 'D%C'
2      THEN
3        BEGIN
3
3         ! set up the pointer to begining of tape owner field
3
3         P = VOL1[VL1$T_VOLOWNER] + 3;
3
3         ! test for encoding
3
3         IF .(.P)<0, 8> NEQ ' '
3         THEN
4             BEGIN
4
4             ! move the UIC group field from the VOL1 label to the buffer
4
4             CH$MOVE(5, .P, CONV_BUF);
4
4             ! remove overlay encoding
4
4             IF .(.P)<0, 8> GEQ 'A'
4             THEN CONV_BUF<0, 8> = .(.P)<0, 8> - ('A' - '0');
4
4             ! convert to ASCII to binary exit with failure not a VMS tape
4
4             IF NOT LIB$CVT_OTB(5, CONV_BUF, VALUE) THEN RETURN FALSE;
4
4             ! fill in the UIC group field
4
4             VOLUME_UIC<16, 16> = .VALUE<0, 16>;
3             END;
3
3         ! point to UIC member field
3
3         P = .P + 5;
3
3         ! test for encoding
3
3         IF .(.P)<0, 8> NEQ ' '
3         THEN
4             BEGIN
```

```
        ! move member number into convert buffer

        CH$MOVE(5, .P, CONV_BUF);

        ! remove overlay encoding

        IF .(.P)<0, 8> GEQ 'A'
        THEN CONV_BUF<0, 8> = .(.P)<0, 8> - ('A' - '0');

        ! convert to ASCII to binary exit when failure not a VAX tape

        IF NOT LIB$CVT_OTB(5, CONV_BUF, VALUE)
        THEN
            BEGIN

            ! patch up UIC before returning

            VOLUME_UIC = .PROCESS_UIC;
            RETURN FALSE;
            END;

        ! fill in the UIC member field

        VOLUME_UIC <0, 16> = .VALUE<0, 16>;
        END;

    ! Now tape_prot must be decoded if both group and member are blank then
    ! all privileges granted
    !
    ! pointer to group uic

    P = .P - 5;

    ! if field is not blank, then there is a protection mask

    IF NOT CH$FAIL(CH$FIND_NOT_CH(10, .P, ' '))
    THEN
        BEGIN

        ! any mask means no world write

        VOLUME_PROT[WORLD_WRITE] = 1;

        ! if the 1st char is a digit then no world access

        IF .(.P)<0, 8> LSS 'A'
        THEN VOLUME_PROT[WORLD_READ] = 1;

        ! pointer to member field

        P = .P + 5;

        ! test for group rights.  all spaces means both read and write

        IF NOT CH$FAIL(CH$FIND_NOT_CH(5, .P, ' '))
        THEN
```

```
;  302        0815  5                 BEGIN
;  303        0816  5
;  304        0817  5                 ! write protection against group if non-blank
;  305        0818  5
;  306        0819  5                 VOLUME_PROT[GROUP_WRITE] = 1;
;  307        0820  5
;  308        0821  5                 ! if the 1st char is a digit then no group access
;  309        0822  5
;  310        0823  5                 IF .(.P)<0, 8> LSS 'A'
;  311        0824  5                 THEN VOLUME_PROT[GROUP_READ] = 1;
;  312        0825  5
;  313        0826  4                 END;
;  314        0827  4
;  315        0828  3              END;
;  316        0829  3
;  317        0830  3           END
;  318        0831  3
;  319        0832  3           ! If there is no VMS protection but was pre ANSI Label Standard
;  320        0833  3           ! version 4 and the tape was created on another DEC operating
;  321        0834  3           ! system that is it has D% information. Then require priviledges
;  322        0835  3           ! to mount the tape.
;  323        0836  3
;  324        0837  2           ELSE
;  325        0838  3           BEGIN
;  326        0839  3              IF .(VOL1[VL1$T_VOLOWNER])<0,16> NEQ 'D%'
;  327        0840  3                 THEN RETURN TRUE
;  328        0841  3                 ELSE RETURN FALSE;
;  329        0842  2           END;
;  330        0843  2
;  331        0844  2        RETURN TRUE;
;  332        0845  1        END;                                      ! end of routine TAPE_OWN_PRO



                                              00FC 00000          .ENTRY  TAPE_OWN_PROT, Save R2,R3,R4,R5,R6,R7            ; 0642
                            57 00000000G  00   9E 00002          MOVAB   LIB$CVT_OTB, R7                                  ;
                            5E            0C   C2 00009          SUBL2   #12, SP                                          ;
                            50       10   AC   D0 0000C          MOVL    VOL1, R0                                         ; 0712
                            34       4F   A0   91 00010          CMPB    79(R0), #52                                      ;
                                          78   13 00014          BEQL    9$                                               ;
00432544   8F      25   A0          18    00   ED 00016          CMPZV   #0, #24, 37(R0), #4400452                        ; 0717
                                          03   13 00020          BEQL    1$                                               ;
                                        00A0   31 00022          BRW     12$                                              ;
                            56       28   A0   9E 00025 1$:      MOVAB   40(R0), P                                        ; 0723
                            20            66   91 00029          CMPB    (P), #32                                         ; 0727
                                          23   13 0002C          BEQL    3$                                               ;
                   04   AE          66    05   28 0002E          MOVC3   #5, (P), CONV_BUF                                ; 0733
                             41     8F    66   91 00033          CMPB    (P), #65                                         ; 0737
                                          05   1F 00037          BLSSU   2$                                               ;
                   04   AE          66    11   83 00039          SUBB3   #17, (P), CONV_BUF                               ; 0738
                                          5E   DD 0003E 2$:      PUSHL   SP                                               ; 0742
                                     08   AE   9F 00040          PUSHAB  CONV_BUF                                         ;
                                          05   DD 00043          PUSHL   #5                                               ;
                            67            03   FB 00045          CALLS   #3, LIB$CVT_OTB                                  ;
                            30            50   E9 00048          BLBC    R0, 5$                                           ;
```

```
                                                          I 3
COMLABPROC                                           16-Sep-1984 02:13:52    VAX-11 Bliss-32 V4.0-742                Page  8
V04-000                                              14-Sep-1984 12:46:36    [MTAACP.SRC]COMLABPRC.B32;1                  (2)

     04  BC          10        10        6E  F0 00048        INSV    VALUE, #16, #16, @VOLUIC                      ; 0746
                               56        05  C0 00051 3$:    ADDL2   #5, P                                         ; 0751
                               20        66  91 00054        CMPB    (P), #32                                      ; 0755
                                         28  13 00057        BEQL    7$
               04  AE          66        05  28 00059        MOVC3   #5, (P), CONV_BUF                             ; 0761
                         41    8F        66  91 0005E        CMPB    (P), #65                                      ; 0765
                                         05  1F 00062        BLSSU   4$
               04  AE          66        11  83 00064        SUBB3   #17, (P), CONV_BUF                            ; 0766
                                         5E  DD 00069 4$:    PUSHL   SP                                            ; 0770
                                    08   AE  9F 0006B        PUSHAB  CONV_BUF
                                         05  DD 0006E        PUSHL   #5
                               67        03  FB 00070        CALLS   #3, LIB$CVT_OTB
                               07        50  E8 00073        BLBS    R0, 6$
                         04    BC   0C   AC  D0 00076        MOVL    PROCESS_UIC, @VOLUIC                          ; 0776
                                         54  11 0007B 5$:    BRB     14$                                           ; 0777
                         04    BC        6E  B0 0007D 6$:    MOVW    VALUE, @VOLUIC                                ; 0782
                               56        04  C2 00081 7$:    SUBL2   #4, P                                         ; 0790
                    76         0A        20  3B 00084        SKPC    #32, #10, -(P)                                ; 0794
                                         02  12 00088        BNEQ    8$
                                         51  D4 0008A        CLRL    R1
                                         51  D5 0008C 8$:    TSTL    R1
                                         3D  13 0008E 9$:    BEQL    13$
                         08    BC  2000  8F  A8 00090        BISW2   #8192, @VOLUME_PROT                           ; 0800
                         41    8F        66  91 00096        CMPB    (P), #65                                      ; 0804
                                         06  1E 0009A        BGEQU   10$
                         08    BC  1000  8F  A8 0009C        BISW2   #4096, @VOLUME_PROT                           ; 0805
                               56        05  C0 000A2 10$:   ADDL2   #5, P                                         ; 0809
                    66         05        20  3B 000A5        SKPC    #32, #5, (P)                                  ; 0813
                                         02  12 000A9        BNEQ    11$
                                         51  D4 000AB        CLRL    R1
                                         51  D5 000AD 11$:   TSTL    R1
                                         1C  13 000AF        BEQL    13$
                         08    BC  0200  8F  A8 000B1        BISW2   #512, @VOLUME_PROT                            ; 0819
                         41    8F        66  91 000B7        CMPB    (P), #65                                      ; 0823
                                         10  1E 000BB        BGEQU   13$
                         08    BC  0100  8F  A8 000BD        BISW2   #256, @VOLUME_PROT                            ; 0824
                                         08  11 000C3        BRB     13$                                           ; 0717
                       2544    8F    25  A0  B1 000C5 12$:   CMPW    37(R0), #9540                                 ; 0839
                                         04  13 000CB        BEQL    14$
                               50        01  D0 000CD 13$:   MOVL    #1, R0                                        ; 0844
                                             04 000D0        RET
                                         50  D4 000D1 14$:   CLRL    R0                                            ; 0845
                                             04 000D3        RET

; Routine Size:  212 bytes,    Routine Base:  $CODE$ + 000C


;  333          0846  1
```

```
1 GLOBAL ROUTINE PROCESS_VOL2_LABEL ( VOLUIC, VOLUME_PROT : REF BITVECTOR[%BPVAL],
1                                     PROCESS_UIC, VOL2 ) =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !       This routine determines the tape_owner and protection of the volume.
1 !       It uses the VOL2 label to intepret the VMS specified/formatted
1 !       protection of this volume. This protection used to exist in the
1 !       OWNER IDENTIFIER field of the VOL1 label. We have moved it into
1 !       this label because of changes which will be adopted in the
1 !       upcoming (version 4) ANSI MAGNETIC TAPE STANDARD
1 !
1 ! CALLING SEQUENCE:
1 !       PROCESS_VOL2_LABEL ( ARG1, ARG2, ARG3, ARG4 )
1 !
1 ! INPUT PARAMETERS:
1 !       ARG1 -  Address of area to store the volume uic
1 !       ARG2 -  Address of area to store the volume protection
1 !       ARG3 -  Process UIC
1 !       ARG4 -  Address of ANSI VOL1 label
1 !
1 ! IMPLICIT INPUTS:
1 !       NONE
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       VOLUME_UIC - owning uic of tape
1 !       VOLUME_PROT -  tape protection
1 !
1 ! ROUTINE VALUE:
1 !       TRUE - Field was blank or was specied in VMS format
1 !       FALSE - Field was non-blank and not VMS format
1 !
1 ! SIDE EFFECTS:
1 !       NONE
1 !
1 ! USER ERRORS:
1 !       NONE
1 !
1 !--
1
2     BEGIN
2
2     BIND
2       VOLUME_UIC = .VOLUIC;                   ! Address of volume uic
2
2     MAP
2       VOL2    : REF BBLOCK;                   ! Address of VOL2 label
2
2     LOCAL
2       CONV_BUF        : VECTOR [6, BYTE],    ! buffer used for converting UIC
2       VALUE,                                  ! used to hold parital UIC's
2       P;                                      ! ptr into VOL2 owner field
2
```

```
2          ! bit numbers for different protections
2          !
2          LITERAL
2              WORLD_WRITE = 13,
2              WORLD_READ  = 12,
2              GROUP_WRITE = 9,
2              GROUP_READ  = 8;
2
2          ! if ANSI tape produced by VAX system, decode tape owner field
2
2          IF .(VOL2[VL2$T_VOLOWNER])<0, 24> EQL 'D%C'
2            THEN
3              BEGIN
3
3              ! set up the pointer to begining of tape owner field
3
3              P = VOL2[VL2$T_VOLOWNER] + 3;
3
3              ! test for encoding
3
3              IF .(.P)<0, 8> NEQ ' '
3              THEN
4                  BEGIN
4
4                  ! move the UIC group field from the VOL2 label to the buffer
4
4                  CH$MOVE(6, .P, CONV_BUF);
4
4                  ! remove overlay encoding
4
4                  IF .(.P)<0, 8> GEQ 'A'
4                  THEN CONV_BUF<0, 8> = .(.P)<0, 8> - ('A' - '0');
4
4                  ! convert to ASCII to binary exit with failure not a VMS tape
4
4                  IF NOT LIB$CVT_OTB(6, CONV_BUF, VALUE) THEN RETURN FALSE;
4
4                  ! fill in the UIC group field
4
4                  VOLUME_UIC<16, 16> = .VALUE<0, 16>;
3                  END;
3
3              ! point to UIC member field
3
3              P = .P + 6;
3
3              ! test for encoding
3
3              IF .(.P)<0, 8> NEQ ' '
3              THEN
4                  BEGIN
4
4                  ! move member number into convert buffer
4
4                  CH$MOVE(6, .P, CONV_BUF);
4
4                  ! remove overlay encoding
```

```

        IF .(.P)<0, 8> GEQ 'A'
        THEN CONV_BUF<0, 8> = .(.P)<0, 8> - ('A' - '0');

        ! convert to ASCII to binary exit when failure not a VAX tape

        IF NOT LIB$CVT_OTB(6, CONV_BUF, VALUE)
        THEN
            BEGIN

            ! patch up UIC before returning

            VOLUME_UIC = .PROCESS_UIC;
            RETURN FALSE;
            END;

        ! fill in the UIC member field

        VOLUME_UIC <0, 16> = .VALUE<0, 16>;
        END;

    ! Now tape_prot must be decoded if both group and member are blank then
    ! all privileges granted
    !
    ! pointer to group uic

    P = .P - 6;

    ! if field is not blank, then there is a protection mask

    IF NOT CH$FAIL(CH$FIND_NOT_CH(12, .P, ' '))
    THEN
        BEGIN

        ! any mask means no world write

        VOLUME_PROT[WORLD_WRITE] = 1;

        ! if the 1st char is a digit then no world access

        IF .(.P)<0, 8> LSS 'A'
        THEN VOLUME_PROT[WORLD_READ] = 1;

        ! pointer to member field

        P = .P + 6;

        ! test for group rights.  all spaces means both read and write

        IF NOT CH$FAIL(CH$FIND_NOT_CH(6, .P, ' '))
        THEN
            BEGIN

            ! write protection against group if non-blank

            VOLUME_PROT[GROUP_WRITE] = 1;
```

```
;  506        1018 5                     ! if the 1st char is a digit then no group access
;  507        1019 5
;  508        1020 5                     IF .(.P)<0, 8> LSS 'A'
;  509        1021 5                     THEN VOLUME_PROT[GROUP_READ] = 1;
;  510        1022 5
;  511        1023 4                     END;
;  512        1024 4
;  513        1025 3                  END;
;  514        1026 3
;  515        1027 2               END;
;  516        1028 2
;  517        1029 2            RETURN TRUE;
;  518        1030 1            END;                                   ! end of routine TAPE_OWN_PRO




                                               00FC 00000        .ENTRY  PROCESS_VOL2_LABEL, Save R2,R3,R4,R5,R6,R7    ; 0847
                          57 00000000G    00     9E 00002        MOVAB   LIB$CVT_OTB, R7                               ;
                          5E              0C     C2 00009        SUBL2   #12, SP                                       ;
                          56         10   AC     D0 0000C        MOVL    VOL2, R6                                      ; 0914
00432544   8F     04   A6   18            00     ED 00010        CMPZV   #0, #24, 4(R6), #4400452                      ;
                                          03     13 0001A        BEQL    1$                                            ;
                                        009D     31 0001C        BRW     10$                                           ;
                          56              07     C0 0001F  1$:   ADDL2   #7, P                                         ; 0920
                          20              66     91 00022        CMPB    (P), #32                                      ; 0924
                                          23     13 00025        BEQL    3$                                            ;
                  04   AE      66         06     28 00027        MOVC3   #6, (P), CONV_BUF                             ; 0930
                          41   8F         66     91 0002C        CMPB    (P), #65                                      ; 0934
                                          05     1F 00030        BLSSU   2$                                            ;
                  04   AE      66         11     83 00032        SUBB3   #17, (P), CONV_BUF                            ; 0935
                                          5E     DD 00037  2$:   PUSHL   SP                                            ; 0939
                                     08   AE     9F 00039        PUSHAB  CONV_BUF                                      ;
                                          06     DD 0003C        PUSHL   #6                                            ;
                          67              03     FB 0003E        CALLS   #3, LIB$CVT_OTB                               ;
                          7C              50     E9 00041        BLBC    R0, 11$                                       ;
      04   BC        10   10              6E     F0 00044        INSV    VALUE, #16, #16, @VOLUIC                      ; 0943
                          56              06     C0 0004A  3$:   ADDL2   #6, P                                         ; 0948
                          20              66     91 0004D        CMPB    (P), #32                                      ; 0952
                                          28     13 00050        BEQL    6$                                            ;
                  04   AE      66         06     28 00052        MOVC3   #6, (P), CONV_BUF                             ; 0958
                          41   8F         66     91 00057        CMPB    (P), #65                                      ; 0962
                                          05     1F 0005B        BLSSU   4$                                            ;
                  04   AE      66         11     83 0005D        SUBB3   #17, (P), CONV_BUF                            ; 0963
                                          5E     DD 00062  4$:   PUSHL   SP                                            ; 0967
                                     08   AE     9F 00064        PUSHAB  CONV_BUF                                      ;
                                          06     DD 00067        PUSHL   #6                                            ;
                          67              03     FB 00069        CALLS   #3, LIB$CVT_OTB                               ;
                          07              50     E8 0006C        BLBS    R0, 5$                                        ;
                          04   BC    0C   AC     D0 0006F        MOVL    PROCESS_UIC, @VOLUIC                          ; 0973
                                          4A     11 00074        BRB     11$                                           ; 0974
                          04   BC         6E     B0 00076  5$:   MOVW    VALUE, @VOLUIC                                ; 0979
                               56         05     C2 0007A  6$:   SUBL2   #5, P                                         ; 0987
                     76        0C         20     3B 0007D        SKPC    #32, #12, -(P)                                ; 0991
                                          02     12 00081        BNEQ    7$                                            ;
                                          51     D4 00083        CLRL    R1                                            ;
                                          51  D5 00085 7$:     TSTL    R1                                  ;
                                          33  13 00087         BEQL    10$                                 ;
                    08   BC      2000     8F  A8 00089         BISW2   #8192, @VOLUME_PROT                 ; 0997
                    41   8F               66  91 0008F         CMPB    (P), #65                            ; 1001
                                          06  1E 00093         BGEQU   8$                                  ;
                    08   BC      1000     8F  A8 00095         BISW2   #4096, @VOLUME_PROT                 ; 1002
                         56               06  C0 0009B 8$:     ADDL2   #6, P                               ; 1006
          66             06               20  3B 0009E         SKPC    #32, #6, (P)                        ; 1010
                                          02  12 000A2         BNEQ    9$                                  ;
                                          51  D4 000A4         CLRL    R1                                  ;
                                          51  D5 000A6 9$:     TSTL    R1                                  ;
                                          12  13 000A8         BEQL    10$                                 ;
                    08   BC      0200     8F  A8 000AA         BISW2   #512, @VOLUME_PROT                  ; 1016
                    41   8F               66  91 000B0         CMPB    (P), #65                            ; 1020
                                          06  1E 000B4         BGEQU   10$                                 ;
                    08   BC      0100     8F  A8 000B6         BISW2   #256, @VOLUME_PROT                  ; 1021
                         50               01  D0 000BC 10$:    MOVL    #1, R0                              ; 1029
                                              04 000BF         RET                                         ;
                                          50  D4 000C0 11$:    CLRL    R0                                  ; 1030
                                              04 000C2         RET                                         ;

; Routine Size:  195 bytes,    Routine Base:  $CODE$ + 00E0
```

```
GLOBAL ROUTINE CHECK_PROT(VOL_PROT,VOL_UIC, PROCUIC,WRT_RING) =

!++
!
! FUNCTIONAL DESCRIPTION:
!       this routine check VMS volume protection
!
! CALLING SEQUENCE:
!       CHECK_PROT(ARG1,ARG2,ARG3,ARG4)
!
! INPUT PARAMETERS:
!       ARG1 - volume protection
!       ARG2 - volume owner UIC
!       ARG3 - Process UIC
!       ARG4 - Write ring status
!
! IMPLICIT INPUTS:
!       NONE
!
! OUTPUT PARAMETERS:
!       NONE
!
! IMPLICIT OUTPUTS:
!       NONE
!
! ROUTINE VALUE:
!       TRUE  - if passes protection
!       FALSE - if does not pass protection
!
! SIDE EFFECTS:
!       NONE
!
! USER ERRORS:
!       NONE
!
!--

    BEGIN

    LOCAL
        PROCESS_UIC     : VECTOR [ 2, WORD ],   ! the process UIC
        WRITE_RING      : BITVECTOR [ 1 ];      ! is this a write mount

    MAP
        VOL_PROT        : REF BITVECTOR,
        VOL_UIC         : REF VECTOR [ 2, WORD ],
        WRT_RING        : BITVECTOR [ 1 ];      ! is this a write mount

    EXTERNAL
        EXE$GL_SYSUIC   : REF BBLOCK ADDRESSING_MODE ( ABSOLUTE );

    LITERAL
        NOT_GROUP_READ  = 8,   ! the group read disable bit
        NOT_GROUP_WRITE = 9,   ! the group write disable bit
        NOT_WORLD_READ  = 12,  ! the world read disable bit
        NOT_WORLD_WRITE = 13;  ! the world write disable bit
```

```
;  577        1088  2      ! get the process UIC
;  578        1089  2
;  579        1090  2      PROCESS_UIC <0,32> = .PROCUIC;
;  580        1091  2
;  581        1092  2      ! get the write protectio of teh tape
;  582        1093  2
;  583        1094  2      WRITE_RING [0] = NOT .WRT_RING [0];
;  584        1095  2
;  585        1096  2      !  check if the user has write access to the tape
;  586        1097  2
;  587        1098  2      IF ( .PROCESS_UIC [ 1 ] LEQ .EXE$GL_SYSUIC ) OR    ! the user's UIC has a
;  588        1099  2                                                         ! system group number
;  589        1100  2
;  590        1101  2         ( NOT .VOL_PROT [ NOT_WORLD_WRITE ] ) OR        ! the tape is world write
;  591        1102  2
;  592        1103  3         (( NOT .VOL_PROT [ NOT_WORLD_READ] ) AND        ! tape is world read and
;  593        1104  2          ( NOT .WRITE_RING [ 0 ]       )) OR            !  read only mount
;  594        1105  2
;  595        1106  3         (( .PROCESS_UIC [ 1 ] EQL .VOL_UIC [ 1 ] ) AND ! (tape's and user's
;  596        1107  4          (( NOT .VOL_PROT [ NOT_GROUP_WRITE ] ) OR      ! group match) and
;  597        1108  5           (( NOT .VOL_PROT [ NOT_GROUP_READ ] ) AND     ! ((tape is group write)
;  598        1109  4            ( NOT .WRITE_RING [ 0 ]           ))OR      ! or (tape is group read
;  599        1110  4                                                        !  and read only mount)
;  600        1111  3            ( .PROCESS_UIC [ 0 ] EQL .VOL_UIC [ 0 ] ))) ! or (member UIC match))
;  601        1112  3
;  602        1113  2         THEN RETURN TRUE;
;  603        1114  2
;  604        1115  3      IF (( .VOL_PROT [ NOT_WORLD_WRITE] ) AND            ! user does not have write
;  605        1116  2          ( NOT .VOL_PROT [ NOT_WORLD_READ])) OR          ! acess but does have read
;  606        1117  2
;  607        1118  3         (( .VOL_PROT [ NOT_GROUP_WRITE]) AND            ! or the same for group
;  608        1119  3          ( NOT .VOL_PROT [ NOT_GROUP_READ]))            ! they have read access
;  609        1120  3
;  610        1121  2          THEN                                           ! Then allow mount but
;  611        1122  3             BEGIN                                       ! set the tape write lock
;  612        1123  3             WRT_RING [ 0 ] = 0;
;  613        1124  3             RETURN TRUE;
;  614        1125  2             END;
;  615        1126  2
;  616        1127  2      !  user does not have needed priviledges return error
;  617        1128  2      !
;  618        1129  2      RETURN FALSE;
;  619        1130  2
;  620        1131  1      END;                                     ! end of Routine CHECK_PROT


                                                               .EXTRN  EXE$GL_SYSUIC

                                               0000 00000      .ENTRY  CHECK_PROT, Save nothing                       ; 1031
                                       0C   AC   DD 00002      PUSHL   PROCUIC                                        ; 1090
          50       10   AC          01      00   EF 00005      EXTZV   #0, #1, WRT_RING, R0                           ; 1094
                                    50      50   92 0000B      MCOMB   R0, R0                                         ;
          51            01          00      50   F0 0000E      INSV    R0, #0, #1, WRITE_RING                         ;
000000006 9F       02   AE          10      00   ED 00013      CMPZV   #0, #16, PROCESS_UIC+2, @#EXE$GL_SYSUIC        ; 1098
                                            42   15 0001D      BLEQ    6$                                             ;
                   3D       04   BC         0D   E1 0001F      BBC     #13, @VOL_PROT, 6$                             ; 1101
          03      04   BC           0C  E0 00024        BBS     #12, @VOL_PROT, 1$              ; 1103
                       35           51  E9 00029        BLBC    WRITE_RING, 6$                  ; 1104
                       50     08    AC  D0 0002C 1$:    MOVL    VOL_UIC, R0                     ; 1106
                  02   A0     02    AE  B1 00030        CMPW    PROCESS_UIC+2, 2(R0)            ;
                                    12  12 00035        BNEQ    3$                              ;
          25      04   BC           09  E1 00037        BBC     #9, @VOL_PROT, 6$               ; 1107
          03      04   BC           08  E0 0003C        BBS     #8, @VOL_PROT, 2$               ; 1108
                       1D           51  E9 00041        BLBC    WRITE_RING, 6$                  ; 1109
                       60           6E  B1 00044 2$:    CMPW    PROCESS_UIC, (R0)               ; 1111
                                    18  13 00047        BEQL    6$                              ;
          05      04   BC           0D  E1 00049 3$:    BBC     #13, @VOL_PROT, 4$              ; 1115
          0A      04   BC           0C  E1 0004E        BBC     #12, @VOL_PROT, 5$              ; 1116
          0D      04   BC           09  E1 00053 4$:    BBC     #9, @VOL_PROT, 7$               ; 1118
          08      04   BC           08  E0 00058        BBS     #8, @VOL_PROT, 7$               ; 1119
                  10   AC           01  8A 0005D 5$:    BICB2   #1, WRT_RING                    ; 1123
                       50           01  D0 00061 6$:    MOVL    #1, R0                          ; 1124
                                        04 00064        RET                                     ;
                                    50  D4 00065 7$:    CLRL    R0                              ; 1129
                                        04 00067        RET                                     ; 1131

; Routine Size:  104 bytes,    Routine Base:  $CODE$ + 01A3


;  621          1132  1
```

```
GLOBAL ROUTINE FORMAT_VOLOWNER(VOL_LABEL,OWNER,PROTECTION) : NOVALUE  =

!++
!
! FUNCTIONAL DESCRIPTION:
!       This routine formats the volume owner field in the VOL2 label
!
! CALLING SEQUENCE:
!       FORMAT_VOLOWNER(ARG1,ARG2,ARG3)
!
! INPUT PARAMETERS:
!       ARG1 - address of VOL2 label
!       ARG2 - owner of tape
!       ARG3 - tape protection
!
! IMPLICIT INPUTS:
!       D%C preinitialized
!
! OUTPUT PARAMETERS:
!       none
!
! IMPLICIT OUTPUTS:
!       none
!
! ROUTINE VALUE:
!       none
!
! SIDE EFFECTS:
!       none
!
! USER ERRORS:
!       none
!
!--

BEGIN

MAP
        VOL_LABEL       : REF BBLOCK,   ! address of VOL1 label
        PROTECTION      : BITVECTOR;    ! protection to be encoded on tape

LOCAL
        DESCR           : VECTOR [2],   ! descriptor
        P;                              ! pointer

LITERAL
        WORLD_WRITE = 13,
        WORLD_READ = 12,
        GROUP_WRITE = 9,
        GROUP_READ = 8;

!
! first convert binary owner to ASCII
!
DESCR[0] = 12;
DESCR[1] = VOL_LABEL[VL2$T_VOLOWNER] + 3;
$FAO(
;  680      P 1190  2           DESCRIPTOR('!6OW!6OW'), 0,
;  681      P 1191  2           DESCR[0],
;  682        1192  2           .OWNER<16,16>,.OWNER<0,16>);
;  683        1193  2
;  684        1194  2 ! now format protection
;  685        1195  2 !
;  686        1196  2 IF NOT .PROTECTION[GROUP_READ] OR NOT .PROTECTION[WORLD_READ] THEN
;  687        1197  3     BEGIN
;  688        1198  3     P = VOL_LABEL[VL2$T_VOLOWNER] + 9;
;  689        1199  3     (.P)<0,8> = .(.P)<0,8> + ('A' - '0');
;  690        1200  2     END;
;  691        1201  2
;  692        1202  2 ! now if group can also write, blank fill member field
;  693        1203  2 !
;  694        1204  2 IF NOT .PROTECTION[GROUP_WRITE] THEN CH$FILL(' ',6,VOL_LABEL[VL2$T_VOLOWNER] + 9);
;  695        1205  2
;  696        1206  2 IF NOT .PROTECTION[WORLD_READ] THEN
;  697        1207  3     BEGIN
;  698        1208  3     P = VOL_LABEL[VL2$T_VOLOWNER] + 3;
;  699        1209  3     (.P)<0,8> = .(.P)<0,8> + ('A' - '0');
;  700        1210  2     END;
;  701        1211  2
;  702        1212  2 IF NOT .PROTECTION[WORLD_WRITE] THEN CH$FILL(' ',12,VOL_LABEL[VL2$T_VOLOWNER] + 3);
;  703        1213  1 END;                                          !end of routine FORMAT_VOLOWNER


                    57  4F  36  21  57  4F  36  21   0020B P.AAB:   .ASCII   \!6OW!6OW\                                   ;
                                                     00213          .BLKB    1
                                          00000008   00214 P.AAA:   .LONG    8                                            ;
                                          00000000'  00218          .ADDRESS P.AAB                                        ;

                                                                    .EXTRN   SYS$FAO

                                              00FC   00000          .ENTRY   FORMAT_VOLOWNER, Save R2,R3,R4,R5,R6,R7      ; 1133
                        5E                04   C2    00002          SUBL2    #4, SP                                       ;
                                          0C   DD    00005          PUSHL    #12                                          ; 1187
                        57          04    AC   D0    00007          MOVL     VOL_LABEL, R7                                ; 1188
                  04    AE          07    A7   9E    0000B          MOVAB    7(R7), DESCR+4                               ;
                        7E          08    AC   3C    00010          MOVZWL   OWNER, -(SP)                                 ; 1192
                        7E          0A    AC   3C    00014          MOVZWL   OWNER+2, -(SP)                               ;
                                    08    AE   9F    00018          PUSHAB   DESCR                                        ;
                                          7E   D4    0001B          CLRL     -(SP)                                        ;
                                    D8    AF   9F    0001D          PUSHAB   P.AAA                                        ;
              00000000G 00                05   FB    00020          CALLS    #5, SYS$FAO                                  ;
                        05          0D    AC   E9    00027          BLBC     PROTECTION+1, 1$                             ; 1196
            07      0D  AC                04   E0    0002B          BBS      #4, PROTECTION+1, 2$                         ;
                        56          0D    A7   9E    00030 1$:      MOVAB    13(R7), P                                    ; 1198
                        66                11   80    00034          ADDB2    #17, (P)                                     ; 1199
            07      0D  AC                01   E0    00037 2$:      BBS      #1, PROTECTION+1, 3$                         ; 1204
    06      20          6E                00   2C    0003C          MOVC5    #0, (SP), #32, #6, 13(R7)                    ;
                                    0D    A7         00041                                                                ;
            07      0D  AC                04   E0    00043 3$:      BBS      #4, PROTECTION+1, 4$                         ; 1206
                        56          07    A7   9E    00048          MOVAB    7(R7), P                                     ; 1208
                        66                11   80    0004C          ADDB2    #17, (P)                                     ; 1209
            07      0D  AC                05   E0    0004F 4$:      BBS      #5, PROTECTION+1, 5$                         ; 1212
    0C      20          6E                00   2C    00054          MOVC5    #0, (SP), #32, #12, 7(R7)                    ;
```

```
                                   07   A7       00059                                                        ;
                                              04 0005B 5$:      RET                                           ; 1213

; Routine Size:  92 bytes,    Routine Base:  $CODE$ + 021C


;  704          1214  1
;  705          1215  1 END
;  706          1216  0 ELUDOM




;                            PSECT SUMMARY
;
;      Name                      Bytes                       Attributes
;
;  $CODE$                             632  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)



;                       Library Statistics
;
;                                          -------- Symbols --------       Pages        Processing
;      File                                Total     Loaded   Percent      Mapped       Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1           18619         13         0      1000           00:01.9




;                            COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:COMLABPRC/OBJ=OBJ$:COMLABPRC MSRC$:COMLABPRC/UPDATE=(ENH$:COMLABPRC)

; Size:         615 code + 17 data bytes
; Run Time:        00:18.2
; Elapsed Time:    00:58.1
; Lines/CPU Min:     4017
; Lexemes/CPU-Min: 26593
; Memory Used:  128 pages
; Compilation Complete
```

0254 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

FREEPG
LIS

DEACCS
LIS

FIND
LIS

DATECN
LIS

FRMOD1
LIS

CREATE
LIS

GETREQ
LIS

EXPIRE
LIS

CNTRL
LIS

FRMHDR
LIS

HEADER
LIS

COMLABPRC
LIS

ENDVOL
LIS

GETFIB
LIS